Московский Государственный Университет Геодезии и Картографии

(МИИГАиК)

факультет геоинформатики и информационной безопасности - информационные системы

Кафедра ИИС

Отчёт по учебной практике

на тему:

«Сервис коротких ссылок с детальной аналитикой переходов»

Работу выполнили:

Студенты 2024-ФГиИб-ИС-1б

Гостев Д.Е.

Быков А.Д.

Студенты 2024-ФГиИБ-ИБ-1б

Колодин Н.Н.

Костоусов Д. А.

Работу проверил:

Калмыков С.А.

Москва 2025

**План работы:**

1. Настройка окружения проекта (Docker, Django, Celery)
2. Создание фронтенд-заготовки на Vue.js
3. Разработка модели угроз (ИБ-направление)
4. Подключение инструмента MobSF
5. Настройка связи клиента с сервером и реализация фильтрации входящих файлов
6. Разработка шаблона отчёта об уязвимостях (ИБ-направление)

**Распределение ролей в команде**

**ИСИТ-направление:**

* Гостев Даниил— frontend (веб-разработка, отвечающая за пользовательский интерфейс и взаимодействие с ним)
* Быков Арсений — backend (реализация серверной логики)

**ИБ-направление:**

* Колодин Никита — Система репутации доменов, помощь с остальными задачами в ИБ-направлении
* Костоусов Дмитрий — Защита от спам-ссылок и вредоносного контента, Rate limiting для предотвращения злоупотреблений, Валидация и санитизация входящих URL

**Как началась работа и как проходила разработка**

Сначала команда разделила зоны ответственности по направлениям (ИСИТ и ИБ). Были выбраны основные инструменты:

* Vs code
* HTML, CSS, JS
* база адресов URLhaus
* FastAPI
* In-Memory хранилище

**Что удалось реализовать**

* Полноценный пользовательский интерфейс (frontend)
* Серверную часть с базовой логикой обработки (backend)
* Связь клиента с сервером
* Система аналитики переходов (геолокация, устройства, время)
* Dashboard для просмотра статистики
* API для интеграции с внешними системами

**Проблемы, с которыми столкнулись**

* Были проблемы с откликами, дизайном и построением графиков.

**Докладчик 1 (Гостев Данил)**

Задача 1: Создание интуитивного интерфейса для сокращения URL

HTML/CSS решение:

- Семантическая разметка форм с правильными типами полей ввода

- Чистая визуальная иерархия с акцентами на основных действиях

- CSS Grid для выравнивания элементов формы

- Плавные переходы для hover-эффектов

- Минималистичная цветовая палитра с контрастными акцентами

JavaScript решение:

- Валидация URL в реальном времени с показом ошибок

- Визуальная обратная связь при операции сокращения

- Управление состоянием кнопок (активно/отключено)

Результат: Пользователи сокращают ссылки за 10-15 секунд с полным пониманием каждого шага процесса

Задача 2: Визуализация многомерной аналитики кликов

HTML/CSS решение:

- Адаптивная CSS Grid сетка 2×2 для графиков

- Карточный дизайн с тенями и скруглениями

- Правильная семантика разделов и заголовков для каждой группы графиков

- Оптимизированные размеры canvas элементов

JavaScript решение:

- Интеграция Chart.js с пользовательскими конфигурациями

- 5 специализированных типов графиков:

- Линейный график для временных рядов

- Кольцевая диаграмма для географического распределения

- Круговая диаграмма для устройств

- Столбчатая диаграмма для источников трафика

- Линейный график для почасовой активности

- Динамическое обновление данных графиков

- Интерактивные подсказки и легенды

Результат: Сложные данные становятся интуитивно понятными через 5 минут изучения панели управления

Задача 3: Обеспечение кроссплатформенной работы

HTML/CSS решение:

- Mobile-first подход в верстке

- 3 ключевых брейкпоинта: 480px, 768px, 1200px

- Flexbox для простых линейных макетов

- CSS Grid для сложных сеток

- Относительные единицы измерения viewport (vw, vh, %)

- Удобные для касания размеры интерактивных элементов (мин. 44px)

JavaScript решение:

- Обнаружение типа устройства и ориентации

- Адаптивная логика для разных сценариев использования

- Оптимизация производительности для мобильных процессоров

Результат: 100% функциональности доступно на устройствах с шириной экрана от 320px до 4K

Задача 4: Создание дизайна

HTML/CSS решение:

- Система CSS-переменных для единообразия

- Последовательная шкала отступов (база 8px)

- Иерархия типографики с четкой визуальной структурой

- Микро-взаимодействия и тонкие анимации

- Готовность к темной/светлой теме

- Доступные цветовые контрасты

JavaScript решение:

- Плавные переходы между состояниями

Результат: Отполированный, неплохой интерфейс

Задача 5: Эффективное управление данными

HTML/CSS решение:

- Семантическая табличная разметка для данных

- Адаптивные таблицы с горизонтальной прокруткой

- Визуальные индикаторы сортировки и фильтрации

- Четкая визуальная иерархия в представлении данных

JavaScript решение:

- Клиентская сортировка по нескольким критериям

- Продвинутая фильтрация с комбинацией условий

- Преобразование данных для экспорта

- JSON сериализация/десериализация

- File API для создания загружаемых экспортов

- Валидация и очистка данных

Результат: Система работы с данными,

Задача 6: Бесшовная навигация между функциями

HTML/CSS решение:

- Единая структура навигации между страницами

- Согласованные компоненты заголовка и подвала

- Четкие визуальные указатели текущего местоположения

- Адаптивные паттерны навигации

JavaScript решение:

- URLSearchParams API для передачи состояния

- Управление сессиями между перезагрузками страниц

- Глубокие ссылки для прямого доступа к аналитике

- Сохранение состояния в URL hash

Результат: Единое приложение, где пользователи не замечают переходов между функциями

**Докладчик 2 (Быков Арсений)**

Моя задача — создать не просто ещё один сокращатель ссылок, а надёжный инструмент, который должен делать три вещи:

\* Быстро создавать короткие ссылки. Пользователь не должен ждать, а его данные — попадать на вредоносные сайты.

\* Мгновенно перенаправлять по ним. Задержка должна быть минимальной, чтобы не терять пользователей.

\* Предоставлять владельцу ссылки подробную аналитику. Чтобы понимать, кто, когда и откуда переходит по его ссылке.

Почему это важно? Потому что современный интернет требует не только удобства, но и безопасности и скорости.

2. РЕШЕНИЕ

Для решения этой задачи я построил бэкенд на FastAPI.

\* Почему FastAPI? Это современный фреймворк, который позволяет обрабатывать тысячи запросов одновременно благодаря асинхронности. Это как многокассовый магазин вместо одного — обслуживание происходит быстрее для всех. Кроме того, он автоматически создаёт красивую и понятную документацию для нашего API, что упрощает жизнь другим разработчикам.

Все данные — ссылки, клики, аналитика — должны храниться и обрабатываться мгновенно.

\* Почему In-Memory хранилище? Я реализовал его по аналогии с Redis. Хранение данных в оперативной памяти — это самый быстрый способ работы с информацией. Редирект по короткой ссылке занимает буквально 5-10 миллисекунд, что пользователь даже не заметит.

3. РЕЗУЛЬТАТ

В итоге я получил готовый к работе высокопроизводительный движок.

\* Производительность: Он стабильно обрабатывает более 1000 запросов в минуту с задержкой менее 50 мс.

\* Надёжность: Даже если что-то идёт не так (например, запрашивают аналитику по несуществующей ссылке), система не падает, а gracefully возвращает демо-данные или понятную ошибку.

\* Готовность к использованию: Сервис полностью готов к развёртыванию и масштабированию. У него есть все необходимые endpoints для здоровья, статистики и удобный дашboard с аналитикой.

Таким образом, мы не просто создали ещё один инструмент, а современный, масштабируемый продукт, решающий реальные задачи пользователей интернета.

За безопасность отвечали мои коллеги с направления ИБ, о чём они сейчас расскажут.

**Докладчик 3 (Колодин Никита)**

Моя зона ответственности — Система репутации доменов

Задача: реализация системы репутации доменов

1. Анализ требований и постановка целей:

Задача: определить, какие параметры влияют на “репутацию” домена и как она будет использоваться в сервисе.

Решение: Определить, что означает «репутация домена» в контексте SecureLink (например, вероятность того, что домен связан с фишингом, спамом и т.д.).

Выделить уровни риска

Определить, где будет использоваться репутация:

1. при генерации коротких ссылок (фильтрация);
2. при переходе пользователей (блокировка/предупреждение);

2. Формирование модели оценки репутации:

Задача: Исключить возможность использования сервиса для спама, фишинга.

Решение: Использовать публичные репутационные API (URLhaus)

Создать весовую систему (если репутация отрицательна, то домен заносится в чс).

3. Интеграция с основным сервисом сокращения ссылок:

Задача: чтобы система влияла на процесс работы SecureLink.

Решение:

Если репутация < порога → запретить сокращение или пометить ссылку как “подозрительную”.

При переходе по ссылке → проверять репутацию в реальном времени.

Если опасный домен → отображать предупреждение пользователю (“Переход может быть небезопасен”).

**Докладчик 4** (Костоусов Дмитрий)

Зона ответственности —Rate Limiting и валидация входящих URL

1. Анализ требований и постановка целей

Задачи:

* Предотвратить злоупотребления сервисом (частые запросы, DoS-атаки).
* Обеспечить безопасность и корректность входящих URL.

Решения:

* Ограничить количество запросов от одного пользователя с помощью rate limiting.
* Добавить строгую валидацию и санитизацию входящих URL для защиты от XSS и некорректных данных.

2. Реализация Rate Limiting для предотвращения злоупотреблений

Задача: ограничить количество запросов от одного клиента за короткий промежуток времени.

Решение:

* Использована библиотека SlowAPI для интеграции лимитов с FastAPI.
* В качестве хранилища для счётчиков запросов выбран Redis.
* Для каждого клиента (по IP-адресу) установлено ограничение 5 запросов в минуту.
* При превышении лимита сервер возвращает ответ 429 Too Many Requests.

Результат:  
Механизм предотвращает автоматизированные атаки, массовые сокращения ссылок и другие формы злоупотреблений, сохраняя стабильную работу сервиса.

3. Валидация и санитизация входящих URL

Задача: не допустить добавления некорректных или потенциально опасных ссылок.

Решение:

* Использована встроенная модель Pydantic BaseModel с типом HttpUrl, проверяющая корректность формата.
* Реализована дополнительная функция is\_valid\_url(url) — проверяет допустимые схемы (http, https), отсекает JavaScript или file-ссылки.
* Все данные проходят проверку до сохранения, предотвращая XSS, SQL-инъекции и ошибки парсинга.

Результат:  
Сервис принимает только безопасные и корректные ссылки, повышая общую надёжность и защищённость системы.

4. Интеграция с основным сервисом сокращения ссылок

Задача: сделать так, чтобы все проверки выполнялись автоматически при каждом запросе.

Решение:

* Проверка URL и репутации выполняется на этапе обработки запроса /shorten.
* Если ссылка признана вредоносной или пользователь превышает лимит запросов — сервис возвращает ошибку.
* Только безопасные ссылки допускаются к генерации коротких URL.

Результат:  
Интеграция с основным API обеспечивает комплексную защиту:  
все входящие ссылки проверяются, фильтруются и ограничиваются по частоте запросов.

**Итог**

Команда успешно реализовала ключевые компоненты сервиса: пользовательский интерфейс, серверную обработку и локальный запуск системы анализа. Несмотря на технические сложности, проект показал работоспособность базовых функций.

Результат практики можно считать успешным: проект функционирует, а студенты получили практический опыт работы с реальными инструментами анализа безопасности, микросервисами и фронтендом/бэкендом в связке.